Developing ActiveX Control

Sahar Muhsen Jaabar
College Of Education, Babylon University
Abed Al-Hamza Mahdi Hamza
College Of Science , University Of Kufa, ,
Esraa Hadi
College of Science For Women, Babylon University

Abstract

One of the most exciting features of Visual languages such as Visual Basic is the ability to create ActiveX documents, forms that can appear within Internet browser windows. ActiveX documents offer built-in view port scrolling, Hyperlinks, and menu negotiation.

We can design ActiveX documents in the same way we design forms. They can contain insert able objects, such as Microsoft Excel pivot tables. They can also show message boxes and secondary forms.

ActiveX documents can also appear in the Microsoft Office Binder, and we can write code to save your document's data in Binder data files.

We can package ActiveX documents in either in-process or out-of-process components.

الخلاصة

واحدة من الخصائص المهمة في اللغات المرئية مثل Visual Basic هي قدرتها على خلق وثائق ActiveX والتي هي عبارة عن النوافذ التي تظهر ضمن نوافذ مستكشف الانترنيت وثائق ActiveX توفر روابط واشرطة تمرير مبنية من الوثيقة . يمكن تصميم وثائق ActiveX بنفس الطريقة التي نصمم بها الاشكال . هذه الوثائق يمكن ان تحتوي على كائنات قابلة للحشر مثل جداول مايكروسوفت اكسل . كذلك يمكن ان تظهر صناديق للرسائل واشكال ثانويه . وثائق ActiveX يمكن ان تظهر ايضا في مايكروسوفت بايندر ويمكن خزن بيانات الوثائق في ملفات البيانات للبايندر ويمكن حزم وثائق ActiveX اما في مكونات داخل العملية او مكونات خارج العملية .

1- What Is an ActiveX Control?(Jeffrey and Francesco , 2003).

An ActiveX component is a unit of executable code, such as an •exe, .dll, or .ocx file, that follows the ActiveX specification for providing objects. ActiveX technology allows programmers to assemble these reusable software components into applications and services.

2-Aim Of This Work

Affording Visual languages developers greater ease, power, and flexibility. ActiveX components are essential tools for building partitioned applications for enterprise-level client/server systems.

3-Termenology

Controls are unlike other objects you create with any visual language like Visual Basic. They're not just code; they have visual parts, like forms — but unlike forms, they can't exist without some kind of container. In addition, controls are used — in different senses — by both developers and end users of applications.

These characteristics of controls require some terminology(Dan , 2002).

3-1 Control Class vs. Control Instance

The control you develop in Visual Basic is actually a control class, a description from which controls will be created. When you put a control on a form. You're creating an instance of this control class.

3-2 Control vs. Control Component

Controls are objects provided by control components, also known as .ocx files. A control component may provide more than one kind of control.

An ActiveX control project contains one or more .ctl files, each of which defines a control class.

3-3 Containers and Sitting

A control instance cannot exist by itself. It must be placed on a container object, such as a form. The process of hooking a control instance up to its container is called sitting — that is, assigning the control a site on the container.

3-4 Interface vs. Appearance

A control consists of three parts, two public and one private. The control's appearance is public, because users see and interact with it. The control's interface — the set of all its properties, methods, and events — is also public, because its used by any program that includes instances of the control.

The private part of a control is its implementation, the code that makes the control work. The effects of a control's implementation can be seen, but the code itself is invisible.

3-5 Author vs. Developer

The author of a control compiles her project as a control component, or .ocx file, which may contain one or more controls. A developer uses the control (or controls) to create an application, and includes the .ocx file in their setup program. The user installs and uses the application.

These terms avoid confusion between the developer of a control and the developer who uses the control in an application.

Developers are not the only direct consumers of ActiveX controls. You can design controls for users to place on documents in desktop applications such as Microsoft Office.

3-6 Design-Time Instance vs. Run-Time Instance

If the project is placed in Run mode, a run-time instance of the control is created when the form is loaded. This run-time instance is destroyed when the form is unloaded. When the form once again appears in design mode, a new design-time instance of the control is created.

4-Creating an ActiveX DLL

Components provide reusable code in the form of objects. An application that uses a component's code, by creating objects and calling their properties and methods, is referred to as a Client (Lan , 2003). .

This section provides step by step instructions on how to define a simple class, and demonstrates the life cycle of objects provided by components. We can use objects created from this class with any application that can use Automation to control objects(JAIN, 2001). .

4-1 Creating the CoffeeMonitor project

1- Start New Project of type **ActiveX DLL called CoffeeMonitor.**

2- Add Module to the project.

3- In the Code window for the module, add the following code:

```
Option Explicit
Public gdatServerStarted As Date
Sub Main()
' Code to be executed when the component starts,
' in response to the first object request. gdatServerStarted = Now
Debug.Print "Executing Sub Main"
End Sub

' Function to provide unique identifiers for objects.

Public Function GetDebugIDQ As Long
Static IngDebugID As Long
IngDebugID = IngDebugID + 1
GetDebugID =IngDebugID
        End Function
```

4- Save Project files , using the following names.

File	File name	Extension
Form	Coffee_TestForm	.frm
Class module	Coffee_CoffeeMonitor	.els
Project	Coffee	.vbp

4-2 Showing Forms from The CoffeeMonitor class

1- open its code window , By double-clicking on CoffeeMonitor.

2- In the Declarations section, add the following Public Enum:

```
Option Explicit

Public Enum feModality

cfeModal = vbModal
cfeModeless =vbModeless End Enum
```

3- Add the following code to the Sub procedure:

```
                    Public Sub ShowForm(Optional Modality As
cfeModality = cfeModal) Dim frm As New TestForm
 If Modality = cfeModeless Then
    frm.Caption == "TestForm - Modeless" Else
  frm.Caption = "TestForm - Modal"
           End If
                              frm.Show Modality
     End Sub
```

4- Click Make Coffee.exe to create a reference executable.

You must put your project in run mode before editing or running the test program, as mentioned in(Catalyst , 2002).

4-3 Providing an Asynchronous Notification Event

One of the most interesting uses for out-of-process components is to provide asynchronous notifications to the client (Road , 2001) . That is, the client doesn't remain blocked while the component executes a method - instead, it goes about its business while the component works on a task or watches for an occurrence of interest.

The procedure in this section sets up a simple asynchronous notification based on a common data processing problem: **How do you know when the coffee is ready?**

The demonstration assumes that you have a coffee maker with a serial interface (however, the demonstration will work even if you don't). The Coffee component tests the serial port periodically to see if the coffee maker's High bit is set, indicating that the coffee is ready.

To set up an asynchronous notification event in the CoffeeMonitor class:

1- Add a Timer control, and set its properties as follows:

Object	Property	Setting
Timer control	(Name)	tmrCoffee
	Enabled	True
	Interval	10000

There's no need to put code in the tmrControl_Timer event procedure. As you'll see, CoffeeMonitor will handle the control's Timer event, test the serial port, and raise the CoffeeReady event to notify CoffeeWatch.

2- In the Declarations section, add the following variables and event declaration:

```
Option Explicit
Private mTestForm As TestForm
Private WithEvents mwtmrCoffee As Timer
Event CoffeeReady()
```

3- (Initialize). Add the following code to create and load an instance of TestForm when the CoffeeMonitor object is created:

```
Private Sub Class_Initialize()
Set mTestForm = New TestForm
Load mTestForm
Set mwtmrCoffee = mTestForm.tmrCoffee End Sub
```

4- In the Procedure drop down, select the Terminate event for the class. Add the following code to the event procedure template:

```
Private Sub Class_Terminate()
Set mwtmrCoffee = Nothing
Unload mTestForm
Set mTestForm = Nothing End Sub
```

5- In the Object drop down, select mwtmrCoffee. The Timer control's only event. Timer, appears in the Procedure drop down, and the event procedure template is added to the code window. Add the following code:

```
Private Sub mwtmrCoffee_Timer()
' (Code to test serial port omitted.)
RaiseEvent CoffeeReady
    End Sub
```

When the CoffeeMonitor object receives the Timer event, it raises its own CoffeeReady event to notify any clients (CoffeeWatch, in this case) that the coffee's ready.

6- Press CTRL+F5 to run the project. Remember, when working with out-of-process components, the component project must be in run mode before you can edit or run the client project.

5- ActiveX EXE Component Creation Summary

In order to introduce new concepts in the most natural order, the procedures in this section have not followed the normal sequence of steps for creating an ActiveX component.

When you create a new ActiveX EXE component, the steps you'll generally follow are these:

1. Determine the features your component will provide.
2. Determine what objects are required to divide the functionality of the component in a logical fashion.
3. Design any forms your component will display.
4. Design the interface — that is, the properties, methods, and events - for each class provided by your component.
5. Create a separate test project, usually a Standard Exe project.
6. Implement the forms required by your component.
7. Implement the interface of each class.
8. As you add each interface element or feature, add features to your test project to exercise the new functionality.
9. Compile your Exe and test it with all potential target applications.

6- Advantages Of Using ActiveX Control

There are many reasons why the ActiveX standard is well-suited for building partitioned components for enterprise-level client/server systems:

Because both distributed COM and Remote Automation technology provide the same programming interface as local Automation, partitioning an application requires little more technical expertise than developing locally executed applications. It also means that ActiveX components can be easily accessed from any client(Measurement Studio -Development Tools for Visual Basic, 2002) .

The algorithms of complex business rules can be stated, commented, and maintained more directly in the language of Visual language than SQL(Catalyst Development, 2002).

ActiveX components can be created in-house (with company-specific business rules implemented) or purchased from third-party vendors (for more generic services) (Dan , 2002).

Deployed on server machines, ActiveX components can remove large computing tasks from the desktop.

Because the binary communication interface between components is identical for both local and remote execution, a component can be moved among machines without recompiling the component or its clients.

Reference

1- Lan R.O. (2003). "Designing Enterprise Applications with Microsoft Visual Basic ".
2- Jeffrey R. and Francesco B. (2003). "Applied Microsoft Frame work programming in Microsoft Visual Basic".
3- Dan F.(2002). "Building Distributed Applications with Visual Basic".
4- Catalyst Development. (2002). "Socket Tools and ActiveX control for Internet Application development.
5- Kayshav Dattatri. "C++ Effective Object-Oriented Software Construction", Prentice Hall PTR, 1997.
6- Road , M. G. (2001) "Mastering Visual Basic 5", HCRO BOOK CENTER, 1997
7- JAIN, A. K., (2001)."The Guide to Building Client/Server Applications with Visual Basic ".
8- "Measurement Studio -Development Tools for Visual Basic(2002) .Visual C# and Visual C++" (@ ai.com/info,2002) .